AUTOMATIC MELODIC GRAMMAR GENERATION FOR POLYPHONIC MUSIC USING A CLASSIFIER SYSTEM

Tsubasa Tanaka
Tokyo University of the Arts
s1311923@fa.geidai.ac.jp

Kiyoshi Furukawa
Tokyo University of the Arts
kf@zkm.de

ABSTRACT

In this paper, we propose a method to generate new melodic styles (melodics) in the automatic composition of polyphonic music. In the proposed method, a melodic style is represented as a grammar that consists of rewriting rules, and the rewriting rules are generated by a classifier system, which is a genetics-based machine learning system. In the previous studies of grammatical approaches, the problem of how to treat polyphony and that of generating new melodic styles automatically haven't been studied very intensively. Therefore, we have chosen to tackle those problems. We modeled the generative process of polyphonic music as asynchronous growth by applying rewriting rules in each voice separately. In addition, we developed a method to automatically generate grammar rules, which are the parameters of the polyphony model. The experimental results show that the proposed method can generate grammar rules and polyphonic music pieces that have characteristic melodic styles.

1. INTRODUCTION

The concept of "automatic" in automatic music composition contains two meanings. One side is a practical and "weak" aspect, which means that computers automate routine works in composition and release humans from them. Realization of musical pieces under given musical forms or theories, or execution of music algorithms that human artists designed, are included in this side. The other is a creative and "strong" aspect, which means that computers do the most artistic parts of works and create novel musical ideas. In this paper, we are interested in the latter. Even in automatic music composition, it is usually humans that design the meta-level structures such as music theories and musical styles. However, the realization of the "strong" automation will be impossible unless such a domain is automatically generated to some extent. Therefore, in this study, we try to cope with the problem of the automatic generation of the meta-level structures.

As a foothold, we focus on melodic grammar (namely, melodics, which is the theory of melody), because melody is one of the most important musical elements. In order to generate satisfying music based on melodic grammars, there are two main problems to be solved. The first problem is how to treat polyphony grammatically. Since polyphony is not a simple linear sequence like text or monophony, it is difficult to formalize. The second problem is how to create melodic styles which don't not exist, yet. Dealing with these two problems, we propose a generic grammatical model for polyphony and the method of generating the grammar rules, which are the parameters of the polyphony model.

To make things clearer, we describe about the problem setting, here. We define polyphonic music as the multiple monophonic melodies played simultaneously satisfying some regulations between the melodies. The regulations are designed and programmed by a human, and they are not generated by the system. The assumed performers are some human keyboardists or a player piano. A music piece is represented by a set of lists for the respective voices. The element of the list is a note that consists of the scale degree and the duration.

The outline of this paper is as follows: Section 2 describes the relationships between previous studies and our study. Section 3 explains how to generate a melody by rewriting rules and the significance of rewriting rules in the actual compositional process. Section 4 presents the polyphony model, dealing with the first problem (treatment of polyphony). Section 5 presents the method of generating grammatical rules, dealing with the second problem (generation of unknown styles). Section 6 describes experimental results by the proposed method. Section 7 summarizes this paper and describes future tasks.

2. RELATED WORK

There are many previous studies that try to find out grammar in music. Generative grammar, which was proposed by linguist Noam Chomsky, has an especially great influence. There are many applications of generative grammar to the field of music, such as GTTM, which connected musicologist Schenker's hierarchical music analysis and generative grammar, the modeling of chord progressions in jazz, and the modeling of folk songs. Such studies are surveyed in [1].

Among the grammatical approaches putting the emphasis on composition, studies of melody generation using L-Systems have been actively carried out [2–5]. L-System is a grammar model originally devised to provide a mathematical model of cell development and plant topology. It can be used to model development of various things. It encodes the object to a symbol string, applies rewrit-

ing rules to the symbol string repetitively, and makes it grow. L-System is considered to be a kind of generative grammar without distinction between terminal and non-terminal symbols. Many variations of the basic L-System such as stochastic, context-sensitive, and parametric grammars have been devised [1]. Because of their advantages, such as simplicity, rapid computing time, and versatility, L-Systems and rewriting rules are being studied actively as procedural technologies that rapidly generate virtual landscapes, appearances of buildings, and so on, especially from recent rise of demand for computer graphics [6, 7].

Other than those, studies of grammatical inference are related to our study. Grammatical inference tries to extract the rules of grammar from a training dataset. For example, [8] is such a study. It tries to find probabilistic grammar from jazz improvisation melodies. Although such studies have something in common with us, our study differs in that they try to imitate some styles of existing melodies depending on training data. We don't try to imitate existing melodic styles, but try to generate new melodic styles.

Overviewing these studies of grammatical approaches, it seems that there is a problem that has not fully been solved yet. That is the problem of how to treat polyphony. Although such forms as the tree-structure of generative grammar, symbol string of L-system, and Markov sequence are suitable for expressing monophonies or chord progressions, it is difficult to treat polyphony, which is a complicated, music-specific structure. That may be the reason that many studies treat only monophony. Although there are examples of rewriting rules that represent polyphony in [3], states of all voices are put together and they are not treated independently. Such a method is not sufficient, because it brings about the possibility that independence between melodies is spoiled or that the number of states becomes too large, as discussed later. To solve this problem, a new model is required.

Other than grammatical approach, there are several types of approaches for polyphonic music generation, such as rule-based approach, stochastic approach, and so on. [9] takes a rule-based method that searches for solutions that satisfy the rules of counterpoint. [10] builds a stochastic model of contrapuntal composition and trains the model by statistical learning. The purpose of these studies is to realize music pieces under classic music theory. Therefore, our study differs from these studies in that our purpose is to generate a music theory itself.

A classifier system (CS), used for generating grammar rules in this study, is a system that generates a set of if-then rules by genetic algorithm (GA) [11]. For example, CS is applied to generate the rules for designing visual shapes [12]. Concerning the application of GA to music grammars, there is a study that uses a genetic method for generating melodic grammars [5]. However, in this study, fitness function is not defined and GA is not used to search grammer rules. Only the genetic operations (mutation and crossover) are used to generate variations of rewriting rules of L-System. Applying GA to search melodic grammar is regarded as a future task, and it is included in our work.

The originality of the method that we present in this paper is at the following three points: (1) Rationalization of the grammatical approach toward melodics on the basis of composer's actual compositional process, rather than on the analogy with languages, music analyses, and the fractal nature of L-Systems. (2) Construction of a model that bridges a gap between grammatical approach and polyphony. (3) Application of CS to generate new melodic grammars.

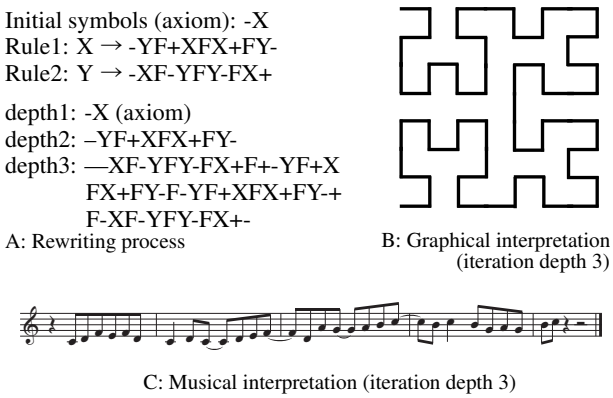

Figure 1. Hilbert curve and its musical interpretation.

3. REWRITING RULES OF MELODY

3.1 Rewriting Rules

A rewriting rule is a production rule used in grammars such as generative grammars and L-Systems and has been used to represent the growth process of various things. As shown in Figure 1-A (upper), a rewriting rule replaces the left symbol string called "predecessor" with the right symbol string called "successor". Applying rewriting rules to the symbol string called "Axiom" which is given initially, the next symbol string is generated, as shown in Figure 1-A (lower). By repeating the application of the same rules to the new symbol string, the symbol string is grown gradually. Figure 1-B shows the third generation of symbol string interpreted graphically. This figure is called "Hilbert curve," which was described by a mathematician Hilbert as a space-filling curve. In this interpretation, "F" means "draw the line forward," "$-$" means "turn left 90 degrees," and "$+$" means "turn right 90 degrees." "X" and "Y" are disregarded in the interpretation phase. Moreover, Prusinkiewicz [2] interpreted the Hilbert curve of the Figure 1-B as a melody (Figure 1-C). Here, a movement along the X-axis means "play a note" and a movement along the Y-axis means movement on a musical scale. The starting point is the lower left. Thus, rewriting rules can be used as grammar rules for generating melodies.

3.2 Melodics and Rewriting Rules

Because the melody that has been shown in the previous subsection was generated by mapping sounds to a graphical figure, the importance of rewriting rules for music or composition is not necessarily clear. To clarify it, we mention the relation between rewriting rules and the compositional process of melodies.

Olivier Messiaen, who is one of the greatest composers

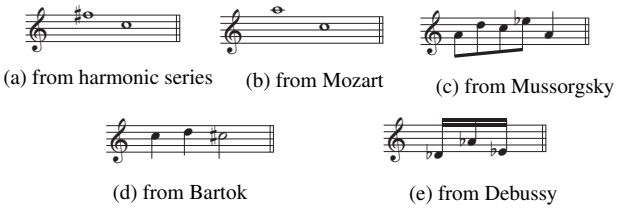

Figure 2. Messiaen's favorite short figures.

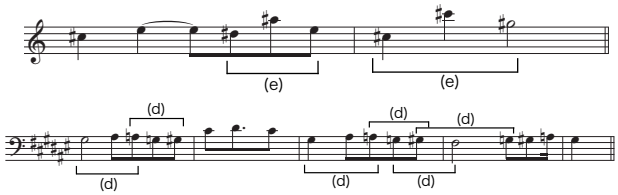

Figure 3. Examples of Messiaen's works in which the short figures are used. Upper: "Arc-en-ciel d'innocence." Lower: "Les Mages."

of the 20th century, wrote about his composition technique in "The Technique of My Musical Language," [13] giving many examples of his own works. We pay attention to chapter 8, which corresponds to the chapter on melodics. Thinking of other composers' idioms, folk songs, Gregorian chants, etc., Messiaen explains how to use characteristic short figures (Figure 2), which are derived from those sources in his melody creation. Those figures are used in Messiaen's own works, like the manner shown in Figure 3.

In these examples, one short figure is frequently used in one melody, and the melody is characterized by it. Furthermore, Messiaen shows how to compound short figures and produce bigger figures. Figure 4 shows such a compound figure and an example in which the compound figure is actually used. Such a composition method shares something in common with rewriting rules. The compound figure of Figure 4 (upper) can be interpreted to be the result of a procedure that uses three rewriting rules $R_1 \sim R_3$, as shown in Figure 5 (lower). Here, R_1 is a rule derived from (a) of Figure 2 and R_2 is derived from (d) of Figure 2 . R_3 is the inversion of R_1. Thus, Messiaen's melody-making method can be naturally interpreted from a viewpoint of rewriting rules. We adopt this method as melodics and generate melodies by applying rewriting rules one after another. Since the same rules are used repeatedly, we can expect that this model generates easily recognizable patterns. However, we can also expect that the model prevents mechanical repetitions and generates organic melodies by compounding multiple figures. Thus, this model would be a good model of the actual compositional process.

4. ASYNCHRONOUS MODEL FOR POLYPHONY

This section discusses the extension of the monophonic model described in the previous section to a polyphonic model. In section 4.1, we think of a naive extension model that simply corrects the contents of the predecessor and the successor of the rewriting rule to a polyphonic data structure. Then we show that such a model may cause difficulties. In section 4.2, we propose a new polyphony model to

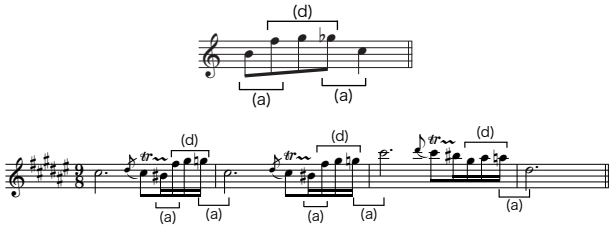

Figure 4. A compound figure (upper) and its use in Messiaen's work, "Chant d'extase dans un paysage triste" (lower).

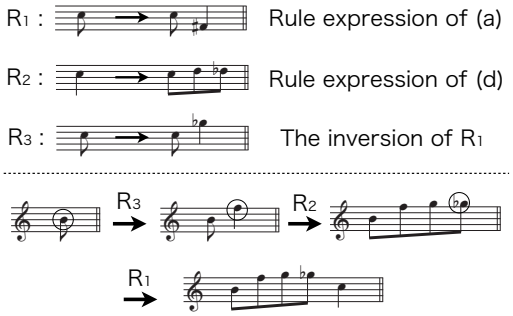

Figure 5. Representation of the figures and their compound using rewriting rules.

avoid the difficulties.

4.1 Difficulty of Naive Extention

In the previous study [3], a representation of polyphony by rewriting rules is already described. In the study, the following example is shown:

$$(CE)|(GC) \rightarrow D(CE)$$

Here, the notes in parentheses mean "play the notes simultaneously." "|" in the predecessor represents a context that the chord (GC) follows after the chord (CE). Since notes in parentheses are always played simultaneously, polyphony with independent rhythms cannot be treated by this notation. In order to extend that representation naively to treat independent rhythms, it is necessary to add the information of durations other than the information of pitchs, and to represent the predecessor and the successor in a following kind of form (here, we describe only the case of two voices):

$$\begin{bmatrix} (p_{11}, d_{11}), & (p_{12}, d_{12}), & \ldots\ldots & , (p_{1n}, d_{1n}) \\ (p_{21}, d_{21}), & (p_{22}, d_{22}), & \ldots & , (p_{2m}, d_{2m}) \end{bmatrix}$$

Here, p_{ij} is the pitch of the jth note in a ith voice, and d_{ij} is the duration of the j th note in the i th voice. p_{ij} can also take a rest symbol, "r," or a "don't care" symbol, "#," which permits any pitch. The next condition, that the sums of the note lengths in respective voices are equal, would be imposed as a time relation between voices:

$$\sum_{k=1}^{n} d_{1k} = \sum_{l=1}^{m} d_{2l}$$

The problem in such a naive extension is the increase of the number of notes within rewriting rules associated with polyphonic rhythms. If we adopt such a formalization, the

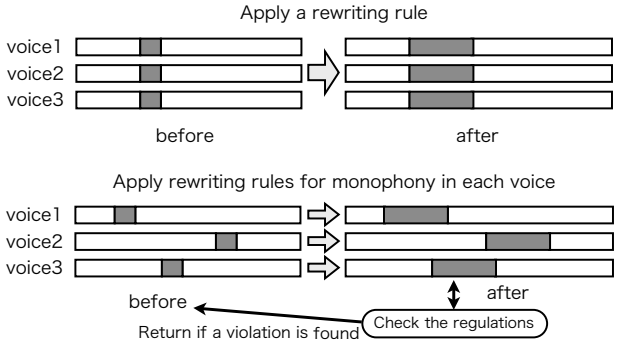

Figure 6. Naive extension (upper) and proposed asynchronous model (lower).

number of required rules will increase exponentially with the increased number of voices. If the number of rules corresponding to the combination of notes in both voices is not sufficient, no rule becomes applicable after a while and that makes it impossible for the piece of music to grow anymore. If we restrict the number of notes in the predecessor to prevent that, the frequency of the use of the same rules will increase and the risk that independence between voices, which is very important for polyphonic music, is spoiled will increase. Moreover, in this notation, rules will become mere enumerations of the combination of the rules for respective voices, and the significance and readability of each rule will become weak.

4.2 Proposed Asynchronous Model

In order to avoid the difficulty of the naive extension model, which rewrites information of all voices simultaneously, we propose a polyphony model that applies the rewriting rules for monophony to respective voices asynchronously. Figure 6 shows the frameworks of both the naive extension model and the proposed asynchronous model.

One important procedure is introduced. Applying rules separately causes unconformities between voices. To prevent such unconformities and maintain polyphony, we introduce ex-post regulations that check the relations between voices. In this paper, we place the following four regulations (the target of automatic generation is purely the melodics within each voice):

- Prohibit voice crossing to avoid registral confusion.
- Set the maximum and minimum limits of the register, which are shared by all the voices.
- Place onsets at the beginning of the measures by more than half of the voices to maintain the metric structure.
- Prohibit the simultaneous use of the same rhythm patterns to maintain the independence of the voices.

Under this proposed model, a piece of music is generated by repeating (T times) the three steps in Figure 7 until it becomes the desired length, after giving initial seeds (axiom) and rewriting rules. Here, We fix the length of successor in each rewriting rule as L (a constant) plus the length of its predecessor. That is to adjust the lengths of respective voices and the metrical structure.

T **times iteration** $\{$

step 1: Select a rewriting rule and a position to which the rule is applied randomly in each voice.

step 2: Apply the selected rule to the selected position once in each voice. After this procedure, every voice grows to be the same length.

step 3: Check the relationship between voices to judge if there is no violation of the regulations. If a violation is found, try again from step 1.

$\}$

Figure 7. Growth algorithm of the proposed model.

After the end of this process, we have a music piece whose length is (initial length $+ L \times T$). This asynchronous model can be considered suitable for polyphony, because it maintains the independence between voices and doesn't restrict the possible combinations between voices too much. In the naive extension model, regulations between voices are reflected in the rewriting rules beforehand. By contrast, in the proposed model, they are to be checked after the application of the rewriting rules. If the regulations between voices are reflected in the rewriting rules, the application of the rules will become easy, in a sense. However, there is a weak point in that the controllable range of the regulations is limited to the range of predecessor. The advantage of the ex-post check is that it can regulate the whole range of the music piece and can compensate for the weak point of the rewriting rule itself. Moreover, the proposed model has a big advantage. Calculated simply, it can reduce the number of rewriting rules from an exponential increase to a linear increase for the number of voices. The number of necessary rules to the number of voices is VR, where V is the number of voices and R is the number of rewriting rules par voice. We set up the number of rewriting rules of respective voices as common number R to simplify the situation. Additionally, C_i is the set of rewriting rules for the ith voice $(1 \leq i \leq V)$, and $c_{ij} (\in C_i)$ is the jth rule for the ith voice $(1 \leq j \leq R)$. In general, C_i differs with respective i. Therefore, the melodic styles of respective voices generally differ from each other.

By the way, the method that composes voices one by one is also possible. Such method can compute efficiently. However, we didn't adopt it because it has an inequality in that the voices composed later are strongly subject to the regulations. That is not suitable for polyphony.

5. GRAMMER GENERAION METHOD

We presented a generic model for polyphony in the previous section. The next problem is to generate grammar rules as the parameters of the proposed model. This section presents a method to generate rewriting rules using classifier system (CS) [11], which is a genetic-based learning system for rule acquisition.

5.1 Classifier System

A classifier system, which was devised by Holland (who is also famous for genetic algorithm), is a system that acquires a set of advantageous rules through adaptation to

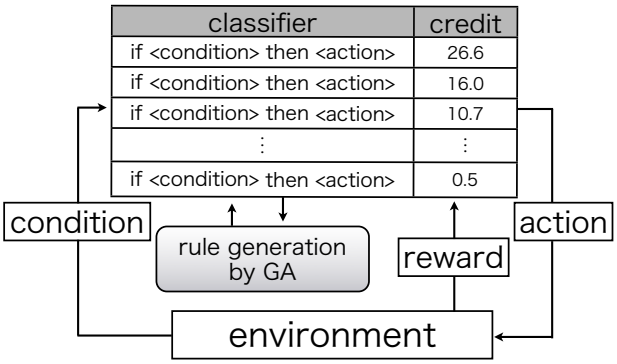

Figure 8. Classifier system.

its environment. It has a set of rules called classifiers on which it takes actions. It receives high rewards from environment after taking actions that had good influences on the environment. Based on the rewards, it learns which rules are beneficial. Furthermore, the classifiers that received low rewards are replaced with new classifiers generated by the genetic algorithm (GA). We can expect an emergence of worthy rules through this process. Figure 8 shows the framework of CS.

A classifier is a rule that has the form "if ⟨ condition ⟩ then ⟨ action ⟩." When the information from the environment satisfies the condition of a classifier, the system sends out the corresponding action to the environment. The numerical value called credit is assigned to each classifier, and credits are continuously updated according to the rewards. When there is more than one classifier applicable to the condition, one classifier is stochastically selected depending on the value of its credit. In this study, since the predecessor and the successor of a rewriting rule can be mapped to the condition and the action of a classifier, we can consider that CS is a suitable method.

5.2 Method for Evaluation of Melodics

In order to apply CS in this study, it is necessary to develop a method for evaluation of melodics and to define the reward, which is given to the credits of rewriting rules. To do so, the following matters should be taken into account:

(1) To use a small number of rules frequently is effective to create characteristics of melodies (see section 3.2).

(2) However, if the number of rules are small, a lack of applicable rule will occur, and that makes further growth of the music piece impossible. Therefore, we should highly appreciate the applicable rules and the rules that are helpful to make the other rules applicable. We can expect that such a valuable rule can be found by measuring its contribution to the growth of the music piece (i.e. how frequently a rule is used).

(3) If the shape of the rules are directed by an artificial evaluation function too much, the purpose of the automatic generation of melodic styles will be spoiled.

In consideration of these, we evaluate the rules by the following means.

- Set R (the number of the classifiers of each voice) as small as possible (because of (1). R is fixed before the execution of the algorithm.).

step 1 Set the initial values.

step 2 Grow the piece by applying the rewriting rules repeatedly (the algorithm of Figure 7). If the piece reaches desired length within a certain number of repetitions, the process is ended. Otherwise, go to step 3.

step 3 Update the credits.

step 4 If the length of the piece is longer than the previous maximum, the rule set and the credits are preserved. Otherwise, recall the rules and their credits, which were preserved last time.

step 5 Replace a certain number of the rules using GA and return to step 2.

Figure 9. Grammer generation algorithm.

- Define the reward to a rule as how frequently the rule is used.
- Don't introduce artificial evaluation function about the contents of the rules (because of (3)).

5.3 Application of Classifier system

Here, we apply CS to the asynchronous model of polyphony shown in section 4.2, and detail the algorithm that generates the rewriting rules. Figure 9 shows the algorithm.

The details of each step of the algorithm are as follows: Step 1 is the phase of initialization. We give an seed musical score, generate initial rewriting rules randomly, and initialize the credit of each rewriting rule. In step 2, the piece is grown gradually by three steps of the polyphony model described in the section 4.2. In each growth step of the piece's length, a maximum number of times of trials is set. If the number of the rule application trials exceeds the limit, we regard that further growth by current rules is impossible and move to step 3. A rank-based selection of applied rule is conducted once in respective voices in each growth step. In the case where the length of the piece reaches the desired length as a result of growth, the processes end. Step 3 evaluates the fitnesses of each rule. A rule that contributed to the growth of the piece in step 2 is given a high reward according to the number of times the rule is used. The credit of a rules is updated by $\{$(the number of times the rule is used this time) + (the previous credit of the rule)$\times\gamma\}$. Here, γ is the discount rate and takes a value between 0 and 1. In step 4, the rule set that left the best result in the past is preserved. This is a treatment to avoid the case that the result worsens by a replacement of rules. In step 5, the rule set is updated by replacing a certain number of current rules by rules generated by genetic operations. The rules to be replaced are selected by the ranking of the credit. In GA, we use the genetic operators, a crossover and mutations, shown in Figure 1.

6. EXPERIMENTAL RESULT

6.1 Conditions of the Experiment

This chapter describes the experiment that generates the grammatical rules and music pieces to confirm the validity of the proposed method. The experimental conditions were set as follows: The system was implemented using Ruby on a Macintosh computer (OS 10.6.8, 2.4GHz Intel Core 2 Duo). The number of the voices is 3 ($V = 3$), and the

Crossover
Execute the one-point crossover of the successors of two rules and adjust the durations of the predecessors.

	Mutation
1	Change the pitch of one note in the successor.
2	Exchange two notes in the successor.
3	Exchange the pitches of two notes in the successor.
4	Exchange the durations of two notes in the successor.
5	Delete a note and give another note the duration of the deleted note in the successor.
6	Divide one note into two notes in the successor, keeping the total duration of the successor.
7	Change the duration of a note in the predecessor and change the same amount of duration in a note in the successor.
8	Delete a note in the predecessor and deduct the same duration of the deleted note from a note in the successor.
9	Insert a note in the successor and add the same duration in a note in the predecessor.
10	Generate a new rewriting rule randomly.

Table 1. Genetic operators.

number of the rewriting rules of each voice is 7 ($R = 7$). The meter was set as 2/4, and the targeted length of the piece is 50 beats. The maximum and minimum of the resister are 96 and 36 in MIDI numbers, respectively. Some restrictions to the form of rewriting rules are imposed. The number of the notes in a predecessor is 1 and the total duration of a successor is a quarter note longer than the corresponding predecessor. Rests are not used. Each voice has its own minimum duration unit. Durations of notes in each voice take the multiples of the minimum duration unit for the voice. Polyrhythms are possible by this condition. Pitches and rewriting rules are based on scales. In addition to the usual rewriting rules to be generated, a rule that forms a cadence is introduced. It is a rewriting rule that adds a finalis whose duration is a half note after any neighbor note. Any scale note can be the finalis. The rule of cadence is used to make pauses to the melody and is applied once out of 10 growth steps. The method of applying the cadence rule is the same as that of the usual rewriting rules. However, an synchronous cadence in which all the voices become cadences is added to the last of the piece.

6.2 Results and Observations

Figures 10 and 12 are the examples of the experimental outputs, and Figure 11 shows the generated rewriting rules of the piece of Figure 10. Generation of these examples took several minutes, respectively. First, in both examples, we can clearly recognize the efficacy of the regulations between the voices in the proposed polyphony model. Voice crossing and simultaneous use of the same rhythm are avoided. That enhances the independence of each voice, which is necessary for polyphonic music. Thanks to the regulation of the note onsets at the beginning of measures, metrical structures can be recognized.

Next, we discuss both examples individually. The example of Figure 10 is a case where every voice has the same minimum duration unit (16th note) and the same rewriting rules. Descending figures that consist of neighboring sixteenth notes are especially remarkable in this piece. They are repeated in each voice, and that produces concords be-

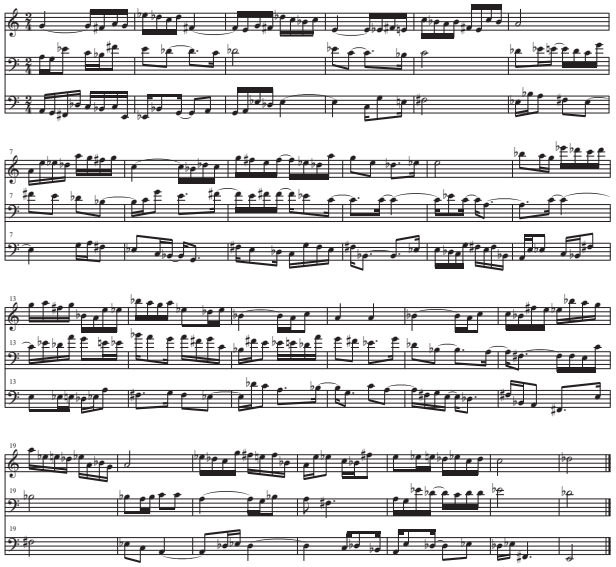

Figure 10. Example 1 (rewriting rules and the minimum duration unit are shared by all the voices).

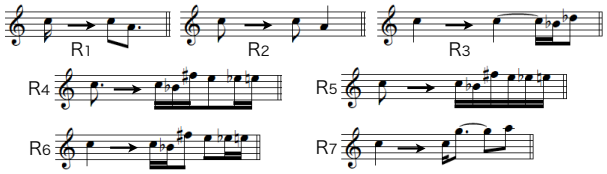

Figure 11. Generated rules of the piece of Figure 10.

tween voices like imitations in contrapuntal music. We can see that the use of common rules can bring about such an effect. On the other hand, the example of Figure 10 is a case of "poly-melodics," in which the voices don't share common rules and a minimum duration unit. In the upper voice, the melody moves intensely with extremely short durations. Here, we can recognize a characteristic rhythm style with dotted notes, which is like the swing of jazz. In the middle voice, although the minimum duration is set as sextuplet sixteenth note, such short durations don't emerge. The melody has a style with gentle rhythms suited for the middle voice and contributes to the stability of harmony and the polyrhythmic effect. In the lower voice, there is an impressive contrapuntal contrast with the upper voice. As a whole, the piece has melodies that have their own melodics and are in harmony with each other.

Figure 13 shows the growth process of the piece of Figure 12. From this graph, we see that the rules couldn't grow the piece in the early stage of GA. However, we could obtain effective rules in the later stage. When we tried the experiment by smaller R than 7, there were many cases where the solution could not be found or the computing time was too long. When R was larger, it was relatively easy to find the solution. In order to maintain the consistency of the melodic style, it is important to make R small. Therefore, it will be significant to find more efficient search methods.

By the way, the emergences of the melodic styles in this experiment were not attributed to the ideas or designs of the authors but to the learning system of CS. This shows that the proposed method has a certain capability to provide new musical ideas. Moreover, the domain of very

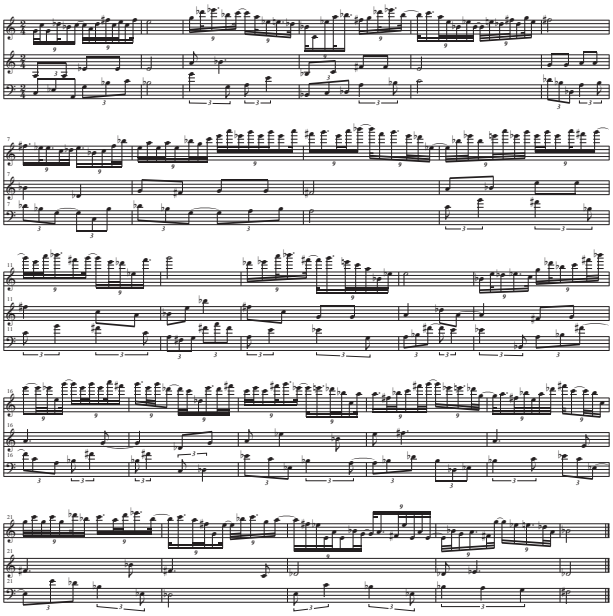

Figure 12. Example 2 (rewriting rules and minimum duration units are not shared between voices).

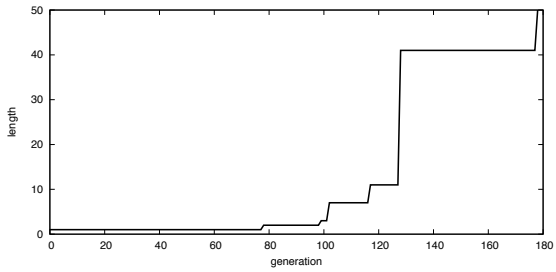

Figure 13. Growth process of the piece of Figure 12.

complicated rhythms like the rhythms of the upper voice of Figure 2 is difficult to treat for human composers. There are few instances that use such complicated rhythms in classic polyphonic music. Therefore, we believe that the proposed method or, in general, the automatic composition itself is significant, because it can explore such difficult domains with relative ease.

7. CONCLUSION

In this paper, we proposed a method of generating melodics for polyphonic music by a grammatical approach. First, we constructed a generic polyphony model that grows respective voices asynchronously. Next, we proposed a method to generate rewriting rules, which are the model parameters. The method is based on a classifier system, which has rarely been used in automatic music composition. From the experimental results, we confirmed that we can acquire rewriting rules that can generate polyphonic music and that the proposed model can regulate the relations between voices appropriately. In the generated music, we observed the emergence of characteristic melodics.

The main future tasks are as follows: Because the proposed method can present readable grammar rules, it may be possible to apply the proposed method to a computer aided composition system, on which humans correct the presented grammar rules and regenerate modified music pieces. Also, we think it is important to deal with not only melodics, but also other music theories such as counterpoint, which regulate the relations between voices, and macroscopic musical forms.

Acknowledgments

This work was supported by Grant-in-Aid for JSPS Fellows.

8. REFERENCES

[1] G. Nierhaus, *Algorithmic Composition*, Springer-Verlag, 2009.

[2] P. Prusinkiewicz, “Score generation with L-systems,” in *Proceedings of the 1986 International Computer Music Conference*, 1986, pp. 455—457.

[3] P. Worth and S. Stepney, “Growing music: musical interpreta tions of L-systems,” In *EvoMUSART workshop, EuroGP 2005, Lausanne, Switzerland, March 2005. LNCS 3449*, Springer, 2005, pp. 545-550.

[4] Jon McCormack, “Grammar-Based Music Composition”. In *Complex systems : from local interactions to global phenomena*, IOS Press, 1996, pp. 321-336.

[5] B. F. Lourenco, J. C. L. Ralha, and M. C. P. Brandao, ”L-Systems, Scores, and Evolutionaly Techniques,” In *Proceedings of 6th Sound and Music Computing Conference*, Porto, 2009.

[6] D. A. Ashlock, S. P. Gent, and K. M. Bryden, “Evolution of L-systems for compact virtual landscape generation,”. in *Proceedings of the 2005 Congress on Evolutionary Computation* Vol. 3, pages 2760—2767, IEEE Press, 2005.

[7] P. Mueller, P. Wonka, S. Haegler, A. Ulmer, L. V. Gool, “Procedural Modeling of Buildings”, In *Proceedings of ACM SIGGRAPH 2006*, pp. 614—623, 2006.

[8] Gillick, J., Tang, K., and Keller, R., “Learning Jazz Grammars,” In *Proceedings of 6th Sound and Music Computing Conference*, Porto, 2009, pp. 125-130.

[9] W. Schottstaedt, “Automatic Counterpoint,” in *Current Directions in Computer Music Reseatrch*, MIT Press, 1989.

[10] T. Tanaka, T. Nishimoto, N. Ono, and S. Sagayama, “Automatic Music Composition Based on Counterpoint and Imitation Using Stochastic Models,” In *Proceedings of 7th Sound and Music Computing Conference*, Barcelona, 2010.

[11] J. H. Holland, *Adaptation in Natural and Artificial Ststems*, MIT press, 1994.

[12] T.Taura, I. Nagasaka, and A.Yamagishi, “Application of evolutionary programming to shape design,” in *Computer-Assisted Design,Vol 30, No.1*, pp. 29-35,1998.

[13] O. Messiaen, *The Technique of My Musical Language*, Translated by J.Satterfield, ALPHONSE LUDEC 1956.